University of Hertfordshire UH

School of Computer Science

BSc Computer Science
Level 6 Direct Entry

Programming Self-Test

Programming Self-Test

This document contains a selection of exercises to help you see if you have sufficient knowledge and skills to complete the on-line programming module at Level 6 in Computer Science.

You can complete the exercises in any language you are familiar with, but it should support both procedural and object-oriented programming. There is no marking or other assistance provided for this test. It is designed to help you assess your own abilities in programming.

How Did I Do?

Sample solutions (in Java, but you can use any language) are given at the end, to help you judge how near/far you were with your own. You should, however, be aware yourself of how easy/difficult these exercises were to complete.
If you found the questions fairly easy, then you should be quite comfortable with the module. If you are not already familiar with Java, you may want to look at the links below, which introduce the language and its concepts.

If you found some of the questions difficult, was this because you have not done some of the ideas before? e.g. are you unfamiliar with procedural or object-oriented programming? Or have you mostly done web scripting or used programming within some other module, such as in Artificial Intelligence? If so, you will probably be able to cope with the module, but should look at the introductory materials to Java indicated below, and practice writing short programs between now and the module start.

If you found the exercises too challenging, then this module is not for you, and you should get advice on an alternative programme of study.

Further Information for Preparation

The programming module is currently run in Java, which is a statically typed language using C-style syntax. You may like to read the following tutorials to become familiar with Java and its syntax, depending on your background (e.g. C# programmers should adjust easily, Javascript programmers will need to focus on the object-oriented aspects, and Python programmers will need to learn a different kind of syntax):

- For Object-Oriented Programming Concepts:
  http://docs.oracle.com/javase/tutorial/java/concepts/index.html
- For the basics of the Java Language (though you can ignore Bitwise/Bit Shift operators):
  http://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html
- The first three parts introducing Classes and Objects:
  http://docs.oracle.com/javase/tutorial/java/javaOO/index.html

Questions

1. Write a program to display all the times tables from 2x to 12x.

2. Write a function/method that takes an array/list of grades (out of 100) and returns a classification depending on the average of the grades.

The classification should be "Fail" for marks < 40%; "C" for marks less than 55; "B" for marks less than 70; and "A" for all higher marks.

Write test code to display each of the results, with at least 4 grades in each list.

3. Create a datatype/class to represent a "Square", with a fixed side.

Write code to display the Square, return its side, and return its area.

Illustrate the class with some test code, creating 10 squares each of side 10, 20, 30 … 100. The squares should be stored in a collection, such as an array or list.

4. Extend your answer to (3) with datatypes/classes for a "Circle" and a "Triangle". Each should contain a method to return their area.
Write some test code creating three examples of each in a single array or list. Then work across the examples displaying the area of each shape.

Sample Solutions

1. You should be using some kind of loop for this, along with a function or two:

```
public class Tables {
 public static void printTableFor (int i) {
  System.out.println ("Printing table for " + i);

  for (int j = 1; j <= 12; j += 1) {
   System.out.println ("" + j + "x" + i + " = " + i*j);
  }

  System.out.println ("----------------------");
 }

 public static void main (String[] args) {
  for (int i = 2; i <= 12; i += 1) {
   printTableFor (i);
  }
 }
}
```

2. Note use of separate functions, cast in getAverage, and if statements:

```
public class Grades {
 public static double getAverage (int[] marks) {
   int sum = 0;

   for (int i = 0; i < marks.length; i += 1) {
    sum = sum + marks[i];
   }

   return (double)sum/marks.length;
 }

 public static String classify (int[] marks) {
  double avg = getAverage (marks);

  if (avg < 40) {
   return "Fail";
  } else if (avg < 55) {
   return "C";
  } else if (avg < 70) {
   return "B";
  } else {
   return "A";
  }
 }

 public static void main (String[] args) {
  int[] marks1 = { 50, 10, 20, 30 };
  System.out.println (classify (marks1));

  int[] marks2 = { 50, 20, 70, 30 };
  System.out.println (classify (marks2));

  int[] marks3 = { 50, 50, 70, 60 };
  System.out.println (classify (marks3));

  int[] marks4 = { 80, 70, 60, 90 };
  System.out.println (classify (marks4));
 }
}
```

3. Create a simple class, and use its methods.

```
public class Square {
 private int side;

 public Square (int side) {
  this.side = side;
 }

 public int getSide () {
  return side;
 }

 public double getArea () {
  return side * side;
 }

 // test method
 public static void main (String[] args) {
  Square[] squares = new Square[10];

  // create some instances
  for (int i = 0; i < squares.length; i += 1) {
   squares[i] = new Square ((i+1) * 10);
  }

  // display the results
  for (int i = 0; i < squares.length; i += 1) {
   System.out.println ("Square " + i +
     " has side " + squares[i].getSide () +
     " and area " + squares[i].getArea ());
  }
 }
}
```

4. Use of parent class, providing contract of getArea

```
public class Shapes {
 // test method
 public static void main (String[] args) {
  Shape[] shapes = new Shape[9];

  // create some instances
  for (int i = 0; i < 3; i += 1) {
   shapes[i] = new Circle ((i+1) * 10);
  }

  for (int i = 3; i < 6; i += 1) {
   shapes[i] = new Square ((i+1) * 10);
  }

  for (int i = 6; i < 9; i += 1) {
   shapes[i] = new Triangle ((i+1) * 10, 30);
  }

  // display the results
  for (int i = 0; i < shapes.length; i += 1) {
   System.out.println ("Shape " + i +
     " has area " + shapes[i].getArea ());
  }
 }
}
```

```
// abstract parent class
abstract class Shape {
 public abstract double getArea (); // guarantees the getArea method
}

class Circle extends Shape {
 private int radius;

 public Circle (int radius) {
  this.radius = radius;
 }

 public int getRadius () {
  return radius;
 }

 public double getArea () {
  return 2 * Math.PI * radius * radius;
 }
}

class Square extends Shape {
 private int side;

 public Square (int side) {
  this.side = side;
 }

 public int getSide () {
  return side;
 }

 public double getArea () {
  return side * side;
 }
}

class Triangle extends Shape {
 private int base, height;

 public Triangle (int base, int height) {
  this.base = base;
  this.height = height;
 }

 public int getBase () {
  return base;
 }

 public int getHeight () {
  return height;
 }

 public double getArea () {
  return 0.5 * base * height;
 }
}
```